

NAME

olvwmrc – Resource file for the OPEN LOOK Virtual Window Manager

SYNOPSIS

\$HOME/.olvwmrc

DESCRIPTION

.olvwmrc is a file which controls advanced keybinding and menu features for the OPEN LOOK Window Manager *olvwm*. Four features of *olvwm* are controlled by entries in this file:

Local Variables

which can be used in key or screen bindings

Key Bindings

which can map specific actions to function keys

Screen Bindings

which can control where certain applications are started

WINMENU

entries which can control the behavior of items selected from the *olvwm* WINMENU menu.

The syntax for each of these entries is given below. Common to all syntax entries is the notion of an “identifier”: this is a string which somehow specifies which window(s) the given entry applies to. When determining if a particular window is affected by a particular rule in *.olvwmrc*, *olvwm* first checks a window’s WM_NAME to see if it matches the identifier listed in the rule. This match is done only for the length of the identifier, so that the identifier *Mail* will match all windows which begin with the 4 letters Mail in their WM_NAME. If this does not match, *olvwm* next checks the instance and then the class fields of the window’s WM_CLASS attribute to check for a possible match. If a match is found for any of these fields, the window is affected by the given rule. Case is significant in checking all matches.

Identifiers may contain any alpha-numeric characters; any other characters must be enclosed within quotes (single or double). Thus, *Mail* is a valid identifier, as is “*Mail Tool*” but *Mail Tool* is not. Similarly, strings which are to be executed should be enclosed in quotes if they contain non alpha-numeric characters. Quotes may be nested in strings, so that to start a clock with the label *foo bar*, you would specify ‘*clock -label “foo bar”*’. Finally, single quotes may be escaped with a backslash, so that the full WM_NAME of DevGuide, for example, should appear as “*OpenWindows Developer’s Guide*”.

All whitespace in this file is ignored; and lines beginning with a # are considered comments (but only if the # is in column 1).

Variables

The *.olvwmrc* file may define local variables by assigning a value to a legal variable name. Variable names must be made up of alphanumeric characters or the ‘_’ character. Variable names may begin with a number and assignments may contain spaces either before or after the ‘=’ operator.

Variable names are referenced by using *%VariableName* or *%{VariableName}*; %% will yield a single %. Variable references may appear as part of the definition of any key or screen binding. Variable references may also appear in assignment statements.

In addition to *olvwm* variables, environment variables may be used in the same contexts using the familiar *\$NAME* or *\${NAME}* syntax; again, \$\$ will yield a single \$.

The following example illustrates the use of variables:

```
#
# Define screen size.
#
Xsize = '1136'
Ysize = '798'

WholeScreenSize = '%{Xsize}x%{Ysize}+3+3'
```

```
#
# Define file names.
#
FileName = '.olvwmrc'

PathName = '$HOME/%FileName'
```

Key/Action Bindings

olvwm can be made to perform a series of actions when a specific key is pressed. The key can be any valid X keysym name and may be specified by itself or with any one or more of the following modifiers: Shift, Control, Alt, Meta, Hyper, Super, Shift Lock, or Caps Lock, in which case the key must be pressed with the given modifiers.

The functionality for a key specified in a binding in *.olvwmrc* takes precedence over any other functions that key might perform. Thus, if you bind the L5 key to an action in *.olvwmrc*, you will not be able to use the L5 key to bring windows to the front; if you bind the R8 key, you will not be able to scroll up on the desktop using that key. Since the unmodified versions of 29 of the possible 35 standard function keys on a type-4 keyboard (L1-L10, F1-F10, and R1-R15) already have a meaning within *olvwm*, it is recommended that at least one modifier be used for keys in this manner so as not to conflict with other key meanings.

There are thirteen valid actions which can be associated with a key:

- Warp** This action requires a single identifier. The youngest window matching this identifier will be located, and the view into the desktop will be warped so that the found window is displayed on the screen. The window itself will not change position relative to the other windows; merely the view into the desktop will be changed. If no matching window is found, the view is unchanged. The mouse is moved into the matching window, and that window is given input focus.
- Open** This action requires a list of identifiers separated by commas. Each iconified window will be matched against this list and those which match any identifier in the list will be opened.
- Close** This action requires a list of identifiers separated by commas. Each non-iconified window will be matched against this list and those which match any identifier in the list will be closed.
- Raise** This action requires a list of identifiers separated by commas. Each window will be matched against this list and those which match any identifier in the list will be raised. Windows will be raised youngest first, so that the oldest windows in the list will end up on top.
- Lower** This action requires a list of identifiers separated by commas. Each window will be matched against this list and those which match any identifier in the list will be lowered. Windows will be lowered youngest first, so that the oldest windows in the list will end up on the bottom.

RaiseLower

This action requires a list of identifiers separated by commas. Each window will be matched against this list and those which match any identifier in the list will be raised to the top of the stack if they are partially obscured or lowered to the bottom of the stack if they are on top.

Execute

This action requires a list of commands separated by commas. Each command will be executed via a Bourne-shell in the same manner as commands given in the *olvwm* menu file [except that multiple commands may be listed in this case.]

- Goto** This action requires a single integer parameter, which is the logical screen to which the desktop should warp when the given key(s) are pressed.
- Quit** This action requires a list of identifiers separated by commas. Each window will be matched against this list and those which match any identifier in the list will be killed.

Geometry

This action requires a single identifier. The identifier must be a valid X geometry string but may be partially specified (may only specify position or size). This geometry will be applied to the current window. If there is no current window this action will have no effect.

Rebind

This action optionally takes a filename parameter. If no parameter is specified the normal search is performed to find the correct version of the *.olvwmrc* file (as at startup). If a parameter is given it is used as the *.olvwmrc* file. All current key bindings are discarded and the *.olvwmrc* file is read. If the *.olvwmrc* files does not exist the current key bindings are not discarded.

Stick

This action requires a single parameter which must be one of the following: *OLVWM_USE_SELECTION*, *on*, *off*, *toggle*, or a list of window names. If the parameter is either *OLVWM_USE_SELECTION* or *toggle*, the sticky attribute of the current window will be toggled. Similarly, if the parameter is a list of window names then those window's sticky attributes will be toggled. The values *on* and *off* can be used to explicitly set the current window's sticky attribute.

SetSize

This action requires a single parameter which must be one of the following: *OLVWM_USE_SELECTION*, *full*, *save*, *store*, *restore*, *toggle*, or a list of window names. If the parameter is *OLVWM_USE_SELECTION* or *toggle*, either the window's current geometry will be saved and its size will be set to full size or its saved geometry will be restored, depending on the window's current state. Similarly, if the parameter is a list of window names then the same action will be performed for those windows. The parameter *save* can be used to preserve the current window's geometry such that a restore size (or *toggle*) will restore the windows position and size. Note that *save* will only store the windows geometry if it has not already been saved. The parameter *store* will always save a windows geometry (possibly overwriting the currently saved geometry). The *restore* parameter will simply restore the current window's saved geometry (if it has one).

Focus

This action requires a single parameter which must be either *save* or *restore*. The *save* parameter will cause the window with focus to be remembered such that a *restore* will restore focus to that window.

These actions may appear in any order and will be performed in the reverse of the order specified. Commands may be listed multiple times; this is useful in case you want a different stacking order than that obtained by using a single raise command. To do this, list separate raise commands for each window and put the raise command for the window you want to be on top first.

The full syntax for a Key/Action binding is

KeyName { Actions }

A Key Name is a valid key (L1-L10, F1-F10, or R1-R15) followed by plus signs and the modifiers desired.

For example, given the following entry:

```
L2 + Shift {
  Warp: "OpenWindows Developer's Guide"
  Execute: '$OPENWINHOME/bin/xview/clock -label "foo bar"',
           "$OPENWINHOME/bin/xview/iconedit"
  Raise: xterm, shelltool
}
```

Then when Shift L2 is pressed, the following will occur:

- 1) The view will shift so that the youngest copy of DevGuide is on the screen.
- 2) A clock will be started; its namestripe will contain foo bar. The IconEditor will also be started.
- 3) All xterms and shelltools will be raised to the front of the stacking order.

Screen Bindings

olvwm can arrange to begin any application relative to a particular logical screen. A "logical screen" is the area on the virtual desktop which maps to the size of your monitor; in the VDM, each logical screen is outlined in dashed lines (unless you've turned this feature off). Screens are numbered by row starting with 1. Note that the position of a logical screen will vary depending on the size of a desktop: in the default (2x3) configuration, screen 4 is in the bottom left-hand corner of the VDM but in a smaller (2x2) configuration, it

is in the bottom right-hand corner.

The syntax for specifying a screen binding is

```
Screen # { Identifiers }
```

where # is the logical number of the screen and *Identifiers* is a list of comma-separated window identifiers for windows which should always start on that screen. Note that it is always possible to move the window to another screen later.

For example, the following entry will ensure that the windows started by Sun's AnswerBook (windows with names Navigator and Viewer) will always start on screen 6:

```
Screen 6 { Navigator, Viewer }
```

WINMENU Actions

When a window is selected in the WINMENU menu, *olvwm* will perform certain actions. The possible actions are the same as those listed above for Key Actions, except that the mouse position will not change on a warp. By default, windows behave as if a warp, raise, and open were performed on the selected window.

To effect a different action list for a particular window, you can specify

```
Identifier { Actions }
```

Each of these is a MenuGroup; one or more of these can appear in the following syntax:

```
WINMENU { MenuGroups }
```

For example, here is a possible entry:

```
WINMENU {
  "File Manager" {
    Warp: "Mail Tool"
    Open: OLVWM_USE_SELECTION
  }
  xterm { }
  "Virtual Desktop" {
    Open: OLVWM_USE_SELECTION
    Execute: "$OPENWINHOME/bin/props"
  }
}
```

If you select the File Manager from your WINMENU, then the view will warp to your Mail Tool instead of your file manager, and your file manager, if closed, will be opened. [This isn't that contrived an example: pretend your file manager is sticky and your mail tool isn't, and you anticipate that you'll need to drag between the two.]

If you select an xterm from your WINMENU, absolutely nothing will happen. This implements a No-Op for that window.

If you select the VDM from your WINMENU, it will be opened and the properties application will be started.

Note that this Identifier list can contain the special entry *OLVWM_USE_SELECTION* which, as you might expect, operates on the single window corresponding to the one you selected. A subtle distinction exists here: given the MenuGroup

```
xterm { Raise: xterm }
```

then ALL xterms will be raised when any xterm is selected via the WINMENU. However, the entry

xterm { Raise: OLVWM_USE_SELECTION }

will raise only the xterm corresponding to the one selected via the WINMENU.

RESOURCES AND KEY BINDINGS

There are a few resources which are particular to the operation of olvwmrc.

VirtualReRead (*boolean*)

When this resource is True, **olvwm** will re-read the *.olvwmrc* file whenever it receives a Function Key event. This will happen whenever a function key is pressed in the VDM or on the root window, or whenever a function key grabbed by **olvwm** is pressed. *Default value: True*

NoVirtualKey (*list of windows*)

This resource disables the virtual keys set up in *.olvwmrc* for a particular window. The list of windows follows the same syntax as other resource lists like MinimalDecor and VirtualSticky. When a window in this list has the input focus and the user executes a key sequence which is mentioned in *.olvwmrc*, that key sequence will be passed to the application rather than initiating the olvwmrc action. Note that this disabling applies only to bindings established via entries in *.olvwmrc*; normal **olvwm** and **olwm** bindings are not affected. *Default value: None*

NoVirtualFKey (*list of windows*)

This resource is like NoVirtualKey, but only the Function keys F1 to F10 will be disabled for the given window. *Default value: None*

NoVirtualLKey (*list of windows*)

This resource is like NoVirtualKey, but only the keys L1 to L10 (which map to F11-F20 on non-Sun keyboards) will be disabled for the given window. *Default value: None*

NoVirtualRKey (*list of windows*)

This resource is like NoVirtualKey, but only the keys R1 to R15 will be disabled for the given window. *Default value: None*

SEE ALSO

olvwm(1), olwm(1)

NOTES

Please see the LEGAL_NOTICES file for full disclosure of copyright information and olvwm(1) for acknowledgments.

BUGS

The multiple interfaces for NoVirtualKeys is something only a Wall Street trader could appreciate.